บุญเลิศ เอี่ยมทัศนา

# TPW
## : งานพิมพ์

***แนะนำเทคนิคการเขียน โปรแกรมการสั่งพิมพ์บน วินโดวส์ให้มีคุณภาพมากยิ่งขึ้น***

มื่อผมเริ่มศึกษาการเขียนโปรแกรมบนวินโดวส์ สิ่งหนึ่งที่ผมต้องการทราบคือ การพิมพ์ออกทางเครื่องพิมพ์ ซึ่งดูเป็นเรื่องยากเย็นมาก เพราะว่าหนังสือภาษาอังกฤษที่เกี่ยวกับการเขียนโปรแกรมบนวินโดวส์ที่ผมมีอยู่สองเล่มไม่ได้เอ่ยถึงเรื่องนี้เลย จนผมได้คู่มือของไมโครซอฟต์มาจึงเป็นเรื่องง่ายมาก

## คู่มือของไมโครซอฟต์

คู่มือของไมโครซอฟต์เพื่อการเขียนโปรแกรมบนวินโดวส์ มีอยู่หลายเล่มด้วยกัน แต่ที่ผมใช้เป็นประจำ คือ

- คู่มือโปรแกรม วินโดวส์ทั้งสองรุ่นใช้ชื่อเดียวกัน คือ Guide to Programming
- คู่มือฟังก์ชัน ถ้าเป็นวินโดวส์ 3.0 ใช้ชื่อว่า Programmer's Reference ซึ่งจะรวมทั้งคู่มือเมสเสจด้วย สำหรับของวินโดวส์ 3.1 คือ Volume 2 : Functions (Volume 1 เป็น Overview)
- คู่มือเมสเสจ ของวินโดวส์ 3.1 แยกต่างหากใช้ชื่อว่า Volume 3 : Messages, Structures และ Macros

สำหรับคำภาษาไทยที่ให้ไว้เป็นหัวข้อ ผมจะนำไปอ้างถึงต่อไปด้วย โดยที่การเขียนบทความชุดนี้ ผมได้นำตัวอย่างโปรแกรมภาษาซีที่เกี่ยวกับคู่มือโปรแกรมมาเขียนด้วยเทอร์โบปาสคาล โดยปรับปรุงตามความต้องการในการใช้งาน

## ที่ว่าง่าย

สิ่งที่ทำให้เป็นเรื่องง่ายก็คือ ดีไวซ์คอนเท็กซ์ (device context) ครับ ทำให้เราใช้เครื่องพิมพ์ได้ง่าย เช่นเดียวกับการนำภาพไปออกจอภาพ แต่การให้กำเนิดดีไวซ์คอนเท็กซ์ของเครื่องพิมพ์ ซึ่งผมจะขอเรียกว่าพรินต์ดีซี มีรายละเอียดมากกว่าเท่านั้น

เช่นในการให้กำเนิดพรินต์ดีซีเพื่อจะใช้กับเครื่องพิมพ์ EPSON LQ-1170 จะต้องผ่านค่าไปให้แก่ CreateDC ดังแสดงในรูปที่ 1 คือ ต้องผ่านชื่อไดร์เวอร์ ชื่อเครื่องพิมพ์ และพอร์ตที่ต่ออยู่ไปให้

## ขั้นตอนของการพิมพ์

เมื่อได้พรินต์ดีซีมาแล้ว ขอให้คิดว่าพรินต์ดีซี คือกระดาษแผ่นหนึ่ง กว้างยาวตามที่ผู้ใช้ติดตั้งไว้ โดยไม่ต้องนึกถึงเครื่องพิมพ์เลย เว้นแต่ที่จะต้องส่งค่าเอสเคปไปให้

การส่งค่าเอสเคปกำหนดให้ส่งด้วยฟังก์ชัน Escape ซึ่งจะเป็นผู้ส่ง

```
PROCEDURE  WndProC.cmPrint(var msg: TMessage);
VAR  device,driver,port : PChar;  prnDC : HDC;
     escres : Integer;  docname : PChar;
BEGIN
  device := 'Epson LQ-1170 ESC/P 2';
  driver := 'ESCP2';  port   := 'LPT1';
  docname := 'TEST PRINT';

  prnDC := CreateDC(driver, device, port, nil);
  escres := Escape(prnDC, StartDoc, StrLen(docname),
                   docname,nil);

  TextOut(prnDC, 20,300, docname, StrLen(docname));
  docname := 'Printing a line of text';
  TextOut(prnDC, 20,50, docname, StrLen(docname));

  Escape(prnDC, NewFrame, 0, nil, nil);
  Escape(prnDC, EndDoc, 0, nil, nil);
  DeleteDC(prnDC);
END;
```

รูปที่ 1 แสดงการเตรียมเครื่องพิมพ์ซึ่งใช้ได้เฉพาะกับเครื่องพิมพ์ EPSON LQ-1170 สำหรับข้อความที่ให้พิมพ์คือข้อความที่อยู่ใน docname

```
PROCEDURE  WndProC.cmPrint(var msg: TMessage);
VAR   device,driver,port : PChar;  prnDC : HDC;
      escres : Integer;  docname : PChar;
      buf : array[0..80] of Char;
BEGIN
  GetProfileString('windows','device', ',,', buf, SizeOf(buf));
  port := GetToken(@buf, ',');  driver := GetToken(buf, ',');
  device := @buf;
  docname := 'TEST PRINT';

  prnDC := CreateDC(driver, device, port, nil);
  escres := Escape(prnDC, StartDoc, StrLen(docname),
                   docname,nil);
  TextOut(prnDC,100,220, device, StrLen(device));
  Rectangle(prnDC,100,50,400,200);
  Ellipse(prnDC,100,50,400,200);

  Escape(prnDC, NewFrame, 0, nil, nil);
  Escape(prnDC, EndDoc, 0, nil, nil);
  DeleteDC(prnDC);
END;
```

รูปที่ 2 แสดงการเตรียมเครื่องพิมพ์ ซึ่งจะเป็นเครื่องพิมพ์ใดก็ได้ที่ติดตั้งอยู่ขณะนั้น

ค่านี้ให้แก่ ไดรเวอร์ของเครื่องพิมพ์ ค่าเอสเคป (ดูได้จาก Printer Escapes ใน Help ของ TPW) เช่น StartDoc จึงเป็นของไดรเวอร์ คือ ของเครื่องพิมพ์นั้น ซึ่งไม่เหมือนกัน แต่มีอยู่จำนวนหนึ่งที่เหมือนกัน ตามที่ผมจะนำมาใช้ในขั้นตอนของการพิมพ์ และก็เพียงพอสำหรับงานพิมพ์โดยทั่วไป ซึ่งมีขั้นตอนดังนี้คือ

1. เตรียมเครื่องพิมพ์ด้วยเอสเคป StartDoc เช่นเดียวกับที่เราใช้คำสั่ง ReWrite เครื่องพิมพ์ในฐานะไฟล์

2. เอาต์พุตไปยังพรินต์ดีซี ซึ่งผมขอให้คิดว่าเป็นกระดาษที่เราจะเขียนหรือวาดภาพด้วยมือของเราเอง คือจะให้เอาต์พุตไปตรงส่วนไหนของกระดาษเมื่อไรก็ได้ ดังเช่นในรูปที่ 1 ผมให้เอาต์พุตบรรทัดล่างก่อนบรรทัดบน และอาจให้วาดภาพกราฟิกได้เช่นกัน

3. ให้นำออกเครื่องพิมพ์ เมื่อเขียนหรือวาดภาพตามต้องการแล้ว ให้นำออกเครื่องพิมพ์ด้วยเอสเคป NewFrame

4. ยุติงานพิมพ์ เมื่อพิมพ์เอกสารตามต้องการแล้ว (ซึ่งเอกสารอาจมีหลายหน้า ดำเนินการตามข้อ 2 และ 3 หลายครั้ง) ให้ยุติงานพิมพ์ด้วยเอสเคป EndDoc เช่นเดียวกับที่ใช้ Close กับเครื่องพิมพ์ในฐานะไฟล์

## พรินต์แมนเนเจอร์

ตามที่ผมกล่าวมานี้ จะเห็นว่าไม่มีส่วนใดเลยที่พาดพิงถึงพรินต์แมนเนเจอร์ แต่เมื่อให้ปฏิบัติตามโปรแกรมนี้ จะปรากฏภาพไอคอนของพรินต์แมนเนเจอร์ (แต่งานพิมพ์จะต้องมากกว่านี้ เพื่อให้มีเวลาพอ) ซึ่งผมเห็นว่าแปลกดี

เมื่อเปิดหน้าต่างนี้ดูจะเห็นชื่อเอกสารตามที่ให้ไว้แก่ docname

## โปรแกรมมาตรฐาน

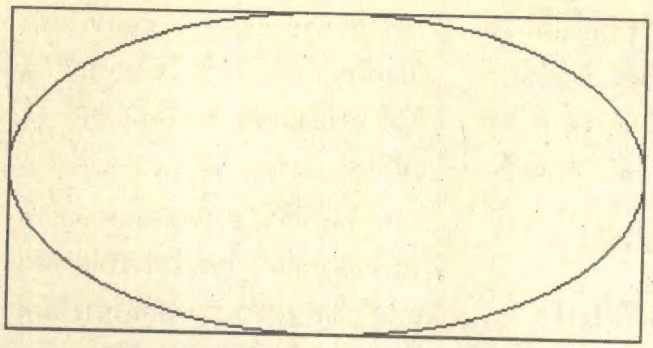
แต่ถ้าหากวินโดวส์ให้เขียนโปรแกรมได้เฉพาะเครื่องพิมพ์หนึ่ง ๆ เท่านั้น ก็ไม่เป็น GDI (Graphics Device Independence) เพื่อการนี้เมื่อผู้ใช้ติดตั้งเครื่องพิมพ์ จะนำค่าที่ผู้ใช้ติดตั้งไปเก็บไว้ในไฟล์ WIN.INI ภายใต้หัวข้อ windows บรรทัดที่มีคำว่า device= และได้เตรียมฟังก์ชัน GetProfile String ไว้ให้เราอ่านค่าตามรูปที่ 2 ซึ่งจะต้องนำมาแยกค่าให้แก่ตัวแปรต่าง ๆ ดังกล่าวมา ด้วยวิธีนี้ทำให้เราไม่ต้องคำนึงว่าผู้ใช้ติดตั้งเครื่องพิมพ์ใดอยู่

ตามโปรแกรมนี้ได้ให้พิมพ์ภาพสี่เหลี่ยม วงรี และข้อความ เพื่อแสดงว่าสามารถให้พิมพ์ผสมกันได้ ดังผลการรันในรูปที่ 3

หากประสงค์จะรันโปรแกรมในรูปที่ 1 และ 2 ให้ทำเมนูมีรายการ Print และเขียนโปรแกรมของสกุล WndProC ที่เหลือกับของ TheApplC ครับ

## สำหรับวินโดวส์ 3.1

สำหรับท่านที่ใช้วินโดวส์ 3.1 ซึ่งหมายถึง จะต้องมีเทอร์โบปาสคาล 1.5 (และต้องคอมพายล์ไฟล์ C:\TPW



รูปที่ 3
ผลของการพิมพ์ภาพกราฟิกกับข้อความผสมกัน

Epson LQ-1170 ESC/P 2

\ WIN31 \ WIN31.PAS เป็นไฟล์ WIN31.TPU ด้วย) ไมโครซอฟต์ได้จัดทำฟังก์ชันให้ใช้แทนค่าเอสเคป และแนะนำให้ใช้ฟังก์ชันเหล่านี้แทน ซึ่งส่วนมากจะใช้ชื่อเดียวกับเอสเคป ดังที่ผมได้นำโปรแกรมในรูปที่ 2 มาเขียนใหม่ในรูปที่ 4 ขอให้เปรียบเทียบดูครับ

สำหรับโปรแกรมในบทความต่อไป (เว้นในรูปที่ 5) จะเขียนโดยใช้เอสเคปไปก่อน

## พิมพ์ภาพบิตแม็พ

การพิมพ์ภาพบิตแม็พ ดำเนินการเช่นเดียวกับการแสดงภาพบิตแม็พ คือ ให้กำเนิดเมมดีซี (memory device context) เป็นดีซีเทียบเท่ากับดีซีของอุปกรณ์ นำแผ่นภาพมาให้ (ซึ่งผมคิดว่าน่าจะใช้คำว่าปะติดมากกว่า) กับเมมดีซีแล้วถ่ายภาพให้แก่ดีซีของอุปกรณ์ในกรณีนี้คือ พรินต์ดีซี (prnDC)

ดังโปรแกรมในรูปที่ 5 ซึ่งผมเขียนตามตัวอย่างในคู่มือโปรแกรมของวินโดวส์ 3.1 อย่างเกือบตรงตัว เว้นแต่การโหลดบิตแม็พมาทำเป็นแผ่นภาพให้แก่ hBMP ผมไม่ได้แสดงไว้ในโปรแกรมนี้ ในส่วนของการพิมพ์มีการดำเนินการดังนี้ครับ

- ขอดีซีจอภาพสำหรับหน้าต่างนี้ให้แก่ wndDC
- ให้กำเนิดดีซีเทียบเท่าจอภาพ ได้แก่ memDC (มีเรื่องที่จะกล่าวต่อไป)
- นำแผ่นภาพบิตแม็พ (hBMP) มาปะติดกับเมมดีซี (memDC) ด้วย SelectObject
- ถ่ายภาพจากเมมดีซีให้แก่พรินต์ดีซี (pmDC) ด้วย BitBlt
- นำออกพิมพ์ด้วย EndPage

```
PROCEDURE  WndProC.cmPrint(var msg: TMessage);
VAR   device,driver,port : PChar;  prnDC : HDC;
      buf : array[0..80] of Char;  docname : PChar;
      escres : Integer;   prnok : Boolean;
      docinfo : TDocInfo; hOldObj : THandle;  memDC,wndDC : HDC;
BEGIN
  GetProfileString('windows','device', ',,', buf, SizeOf(buf));
  port := GetToken(@buf, ',');  driver := GetToken(buf, ',');
  device := @buf;
  docname := 'TEST PRINT';

  prnDC := CreateDC(driver, device, port, nil);
  docinfo.cbSize := SizeOf(docinfo);
  docinfo.lpszDocName := docname;  docinfo.lpszOutput := nil;
  StartDoc(prnDC, docinfo);  StartPage(prnDC);

  TextOut(prnDC,100,320, device, StrLen(device));
  Rectangle(prnDC,100,50,400,200);
  Ellipse(prnDC,100,50,400,200);

  EndPage(prnDC);  EndDoc(prnDC);

  SelectObject(memDC, hOldObj);  DeleteDC(memDC);
  DeleteDC(prnDC);
END;
```

รูปที่ 4 แสดงการพิมพ์โดยใช้ฟังก์ชันแทนเอสเคป

```
PROCEDURE  WndProC.cmPrint(var msg: TMessage);
VAR   device,driver,port : PChar;  prnDC : HDC;
      buf : array[0..80] of Char;  docname : PChar;
      escres : Integer;   prnok : Boolean;
      docinfo : TDocInfo; hOldObj : THandle;  memDC,wndDC : HDC;
BEGIN
  if  hBMP = 0  then  Exit;

  GetProfileString('windows','device', ',,', buf, SizeOf(buf));
  port := GetToken(@buf, ',');  driver := GetToken(buf, ',');
  device := @buf;
  docname := 'TEST PRINT';

  wndDC := GetDC(HWindow);
  prnDC := CreateDC(driver, device, port, nil);
  memDC := CreateCompatibleDC(wndDC);
  ReleaseDC(HWindow, wndDC);
  hOldObj := SelectObject(memDC, hBMP);

  docinfo.cbSize := SizeOf(docinfo);
  docinfo.lpszDocName := docname;  docinfo.lpszOutput := nil;
  StartDoc(prnDC, docinfo);  StartPage(prnDC);

  BitBlt(prnDC, 20,10,wwid,hhi,memDC,0,0, srcCopy);

  EndPage(prnDC);  EndDoc(prnDC);


  SelectObject(memDC, hOldObj);  DeleteDC(memDC);
  DeleteDC(prnDC);
END;
```

รูปที่ 5 โปรแกรมพิมพ์ภาพบิตแม็พ

## ดีซีไม่เทียบเท่า

ผมสงสัยมานานแล้วว่า ตามโปรแกรมในรูปที่ 5 เมื่อให้กำเนิดดีซีเทียบเท่าคือ memDC ของอุปกรณ์ใดก็ควรเป็นของอุปกรณ์นั้น ดังเช่นของเครื่องพิมพ์ก็ควรเป็น CreateCompatibleDC (prnDC) แต่ในคู่มือโปรแกรมของวินโดวส์ 3.0 ให้ใช้ของหน้าต่าง คือ wndDC

ต่อมาในคู่มือโปรแกรมของวินโดวส์ 3.1 ให้ใช้ prnDC ซึ่งผมได้เขียนโปรแกรมตามนี้ ปรากฏว่าไม่เป็นผลครับ จึงต้องมาต่อเติมโปรแกรม GetDC และนำค่า wndDC มาใช้ ดังที่เห็นอยู่ในโปรแกรมขณะนี้

ไม่ทราบเหมือนกันครับว่าเป็นข้อบกพร่องของเทอร์โบปาสคาล หรือของวินโดวส์ หรือของผม ก็ต้องขอให้ผู้ที่เขียนโปรแกรมด้วยภาษาซีทดสอบและถ้ากรุณาแจ้งผลให้ทราบจะขอบคุณมากครับ

## ขนาดของภาพ

ขนาดของภาพบิตแม็พที่พิมพ์ด้วยวิธีนี้จะเล็กกว่าที่พิมพ์ด้วย Paint Brush มาก คือประมาณครึ่งหนึ่ง ซึ่งผมไม่ทราบเหมือนกันว่า ถ้าให้กำเนิดดีซีเทียบเท่าเครื่องพิมพ์ได้ ภาพที่พิมพ์ออกมาจะโตขึ้นหรือไม่

แต่ก็ตรงกับความเป็นจริง เพราะเราพิมพ์ภาพจากจุด ซึ่งถ้าเป็นจอภาพจุดหนึ่ง ๆ คือ จุดภาพหนึ่งบนจอภาพ ขนาดของภาพจึงขึ้นกับขนาดและความละเอียดของจอภาพ

สำหรับเครื่องพิมพ์ตามตัวอย่างที่ยกมา เมื่อตั้งค่าความละเอียด (resolution) ไว้ที่ 180 × 180 จุดต่อนิ้ว ภาพขนาด 640 × 480 จุดคือ เต็มจอภาพจะกว้างเพียง 3.56 นิ้วบนกระดาษ ถ้าเป็นเครื่องพิมพ์เลเซอร์ที่ตั้งค่าความละเอียดไว้ 300 × 300 ก็จะได้ภาพขนาด 2.13 × 16 บนกระดาษ หากต้องการให้ใหญ่กว่านี้ก็ต้องขยายครับ

## โปรแกรมย่อยแสดงหรือพิมพ์ภาพบิตแม็พ

งานพิมพ์ขั้นพื้นฐานมีเพียงเท่านี้ แต่เมื่อจะนำมาเขียนโปรแกรมให้ใช้งานได้ทั่วไป จะมีรายละเอียดอีกพอสมควร ทั้งในส่วนที่เกี่ยวกับงานพิมพ์โดยตรงและเรื่องอื่น ๆ

เรื่องหนึ่งที่เห็นว่าควรทำเป็นโปรแกรมย่อยไว้ก่อน คือการให้ค่าบิตแม็พแก่พรินต์ดีซีหรือดิสเพลย์ดีซีก็ได้ ดังโปรแกรม PrnDspBMPs ในรูปที่ 6

โดยที่ถ้าต้องการให้แสดงภาพทางจอภาพคือ ในหน้าต่างให้ผ่านค่า 0 มาให้แก่ prnDC หากต้องการพิมพ์ต้องให้ค่าพรินต์ดีซีแก่ prnDC จะเป็นการให้ค่าบิตแม็พแก่พรินต์ดีซี เพื่อรอคำสั่งพิมพ์ด้วยเอสเคป EndPage ดังกล่าวมา ตามค่าอื่น ๆ ที่ผ่านมาให้ดังนี้

- hBMP คือแผ่นภาพบิตแม็พ

```
function   GetToken(astr : PChar; at : Char) : PChar;
procedure  DispBMPs(hWndd : HWnd; hBMP : HBitmap;
                    dx,dy,dwid,dhi : Integer; stretch : Boolean;
                    sx,sy,swid,shi : Integer);
procedure  PrnDspBMPs(hWndd : HWnd; hBMP : HBitmap;  prnDC : HDC;
                    dx,dy,dwid,dhi : Integer; stretch : Boolean;
                    sx,sy,swid,shi : Integer);



FUNCTION   GetToken(astr : PChar; at : Char) : PChar;
VAR   p : PChar;
BEGIN
  p := StrRScan(astr,at);
  if  p <> nil  then
    begin  p[0] := #0;  inc(p)  end;
  GetToken := p;
END;


PROCEDURE  PrnDspBMPs(hWndd : HWnd; hBMP : HBitmap;  prnDC : HDC;
                    dx,dy,dwid,dhi : Integer; stretch : Boolean;
                    sx,sy,swid,shi : Integer);
VAR   hOldObj : THandle;  wndDC,memDC,adc : HDC;
BEGIN
  wndDC := GetDC(hWndd);
  memDC := CreateCompatibleDC(wndDC);
  hOldObj := SelectObject(memDC, hBMP);
  if  prnDC = 0  then  adc := wndDC  else  adc :=  prnDC ;
  if  stretch  then
    StretchBlt(adc, dx,dy,dwid,dhi, memDC,sx,sy,swid,shi, srcCopy)
  else
    BitBlt(adc, dx,dy,dwid,dhi, memDC,sx,sy, srcCopy);
  ReleaseDC(hWndd, wndDC);
  SelectObject(memDC, hOldObj);
  DeleteDC(memDC);
END;


PROCEDURE  DispBMPs(hWndd : HWnd; hBMP : HBitmap;
                    dx,dy,dwid,dhi : Integer; stretch : Boolean;
                    sx,sy,swid,shi : Integer);
BEGIN
 PrnDspBMPs(hWndd,hBmp,0,dx,dy,dwid,dhi,stretch,sx,sy,swid,shi);
END;
```

รูปที่ 6 โปรแกรมย่อย PrnDspBMPs เพื่อการพิมพ์ภาพบิตแม็พหรือนำออกแสดงในหน้าต่าง

(ตามที่กล่าวในไมโครฯ 92)

- dx, dy คือจุดมุมบนซ้าย และ dwid กับ dhi คือความกว้างและความสูงของภาพปลายทาง ว่าจะให้อยู่ตรงส่วนใดของหน้าต่างหรือกระดาษ

- sc, sy คือจุดมุมบนซ้าย และ swid กับ shi คือความกว้างและความสูงของภาพต้นทาง ความหมายคือ จะตัดส่วนใดของแผ่นภาพไปให้แก่ปลายทาง

- stretch หากให้เป็นจริงจะเป็นการย่อขยายคือ ถ้าภาพปลายทางเล็กกว่าจะเป็นการย่อ ถ้าใหญ่กว่าจะเป็นการขยาย หากให้เป็นเท็จจะเป็นการลอกคือถ้าภาพปลายทางเล็กกว่าภาพที่เหลือจะถูกขลิบออก ถ้าใหญ่กว่าก็จะไม่เต็มเนื้อที่ปลายทาง

```
    PrinterCP = ^PrinterC;
    PrinterC  = OBJECT(TObject)
      abDlgPtr : AbortDlgCP;
      margs : MarginT;
      pgewid,pgehi : Integer;
      botlne,curlne,lnehi : Integer;
      prnDC : HDC;
      prnok : Boolean;
    constructor  Init(hWndd : HWnd);
    destructor   Done; virtual;
    procedure  PaperSetting;
    procedure  SetMargins(lft,top,bot,dif : Integer);
    function   StartADoc(docname : PChar) : Boolean;
    procedure  EndADoc;
    procedure  PrintEditor(edtr : PEdit);
    procedure  PrintEdBlck(edtr : PEdit);
    procedure  PrintFile(finme : PChar);
    procedure  PrintLine(txt : PChar);
    procedure  PrintLineAt(xx,yy : Integer; txt : PChar);
    procedure  NewLine; virtual;
    function   NewPage : Boolean;
    function   CanPrintBMPs : Boolean;
    procedure  DoSetting;
               END;

procedure  DosPrnTxtFi(finme : PChar);
procedure  DosPrnBlck(edtr : PEdit);
procedure  WrtBlck(edtr : PEdit);
procedure  ReadBlck(edtr : PEdit);


Implementation
```

ลิสติงที่ 1.2

## ไฟล์ไดอะล็อกบ็อกซ์

ในงานพิมพ์สิ่งที่ขาดไม่ได้คือ ไฟล์ไดอะล็อกบ็อกซ์ ให้ผู้ใช้เลือกไฟล์มาพิมพ์ (ไฟล์เดียวหรือหลายไฟล์) ซึ่งผมได้เตรียมการไว้แล้วในยูนิต UFiDlg (ไมโครฯ 91) และได้เตรียมยูนิต Utem ไว้ (ในไมโครฯ 92) เพื่อที่จะรวมกับโปรแกรมในรูปที่ 6 เป็นยูนิต UBMPs

แต่ตอนนี้ขอเปลี่ยนใจครับ เพราะยังมียูนิตของเทอร์โบปาสคาลอีกมากของเราจึงควรให้น้อยยูนิตที่สุด จึงขอยุบยูนิต Utem ไปรวมกับยูนิต UFiDlg รวมทั้งโปรแกรมย่อยตามรูปที่ 6 ด้วย

สำหรับท่านที่ป้อนโปรแกรมมาเป็นลำดับกรุณาดำเนินการตามนี้ด้วยครับ

```
UNIT   UPrint1;

Interface

{$R uprint.res}

USES   WinTypes,WinProcs,WObjects,Strings,ufidlg;

CONST  abdlgres  = 'ABORT_DLG';
       txtdlgres = 'SETTEXT_DLG';
       bmpdlgres = 'SETBMP_DLG';

       id_LeftMar = 101;   id_LineSpace = 104;

TYPE

  MargIndT  = (lftt,topp,bott,lnespcc);
  MarginP   = ^MarginT;
  MarginT   = array[MargIndT] of Integer;

  EdNumArT = array[MargIndT] of EditNumCP;

  AbortDlgCP = ^AbortDlgC;
  AbortDlgC  = OBJECT(TDialog)
procedure  SetupWindow; virtual;
procedure  wmCommand(var msg : TMessage);
             virtual wm_First + wm_Command;
              END;

  PrnSetDlgCP = ^PrnSetDlgC;
  PrnSetDlgC  = OBJECT(TDialog)
    datary    : MarginP;
    ednums    : EdNumArT;
    idfst,idlst : Integer;
constructor  Init(apar : PWindowsObject; atitle : PChar;
                  dataaray : MarginP; fstid,lstid : Integer)
procedure  SetUpWindow; virtual;
procedure  Ok(var msg : TMessage); virtual id first + id_ok;
             END;
```

ลิสติงที่ 1.1

## ยูนิต UPrint1

โปรแกรมสำหรับงานพิมพ์ทั้งหมดผมจะให้อยู่ในยูนิต UPrint1 เพื่อความสะดวกในการนำมาใช้ (ที่ให้มีเลข 1 ต่อท้ายเพราะจะต่อเติมอีก ผมเกรงว่าจะสับสน) ดังที่ได้แสดงส่วน Interface ไว้ในลิสติง 1.1 และ 1.2 มีรายละเอียด

ตามที่จะกล่าวในหัวข้อต่อ ๆ ไป

## ยุติการพิมพ์และข้อขัดข้อง

มีเรื่องที่ผมไม่ได้กล่าวในตอนแรก คือ โปรแกรมที่ดีจะต้องให้ผู้ใช้หยุดงานพิมพ์ได้ด้วย สำหรับการยุติการพิมพ์นี้ขอทำความเข้าใจให้ชัดเจนก่อนครับ คือจะยุติได้ในช่วงที่ใช้คำสั่งเอสเคป NewFrame เท่านั้น (ซึ่งอาจทำงานเป็นวงรอบอยู่) หากเลยไปถึง EndDoc แล้วจะหยุดไม่ได้ ถึงแม้เครื่องจะยังไม่พิมพ์งานของเรา

ทั้งนี้เพราะจะเป็นการส่งงานของเราไปให้แก่พรินต์แมนเนเจอร์เรียบร้อยแล้ว หากพิมพ์งานอื่นอยู่ก็จะไปเข้าคิวอยู่ในพรินต์แมนเนเจอร์ หากต้องการหยุดจะต้องไปยกเลิกในพรินต์แมนเนเจอร์ หรือถ้ากระทันหันก็ต้องที่เครื่องครับ

สำหรับการยุติการพิมพ์ในแอพพลิเคชัน ทางวินโดวส์ได้กำหนดให้ดำเนินการใน 4 เรื่อง ตามที่ได้เขียนโปรแกรมไว้ในลิสติง 1.3 กล่าวคือ

1. จัดตัวแปรไว้เป็นสื่อกลางระหว่างไดอะล็อกบ็อกซ์กับฟังก์ชันยุติการพิมพ์คือ ตัวแปร usrstop ซึ่งต้องให้เป็นโกลบอล (global)

2. จัดไดอะล็อกบ็อกซ์สกุล AbortDlgC ไว้ให้ผู้ใช้ยุติการพิมพ์ดังในรูปที่ 7 มีโปรแกรมอยู่ในลิสติง 1.3 ซึ่งเพียงแต่ให้ค่าโฟกัสแก่ตนเองเมื่อเริ่มต้น และดักรับคอมมานด์เมสเสจ คือ wm_Command เพื่อให้ usrstop เป็นจริง (โดยไม่ตรวจสอบค่าไอดี)

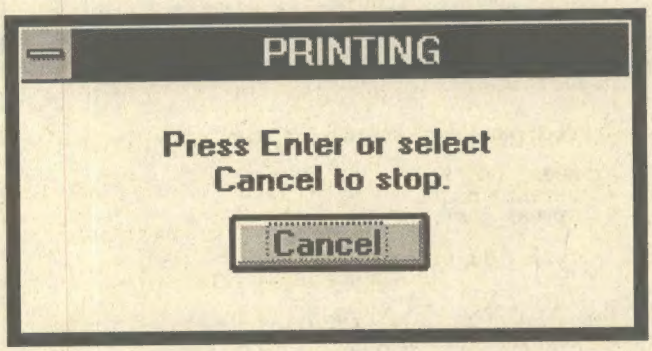
สำหรับไดอะล็อกบ็อกซ์นี้ ในการให้กำเนิด (ตามที่จะกล่าวต่อไป) ต้องให้เป็นโมเดลเลสส์ (modeless dialogbox) เพราะถ้าให้เป็นแบบโมเดล (model dialogbox) การทำงานจะค้างอยู่ที่ไดอะล็อกบ็อกซ์จนกว่าผู้ใช้จะปิดไดอะล็อกบ็อกซ์ แต่ในงานนี้ต้องการให้แสดงไดอะล็อกบ็อกซ์ค้างไว้ ส่วนตัวโปรแกรมจะให้ทำงานอื่นต่อไป

มีเรื่องที่ต้องจำอีกครับคือ ผมดำเนินการตามที่กล่าวในคู่มือโปรแกรมทุกประการ ตรงที่ต้องจำคือในการสร้างรีซอร์สได้ให้ค่าไอดีของปุ่ม Cancel เป็น 2 ตามค่าปกติของมัน ปรากฏว่าไม่แสดงภาพไดอะล็อกบ็อกซ์นี้ ผมต้องใช้เวลาทั้งบ่ายค้นหาในคู่มือ และหนังสือทุกเล่มที่มี ไม่มีเล่มใดเอ่ยถึงเรื่องนี้ (หรือว่าผมหาไม่พบ) ในที่สุดต้องใช้วิธี trial and error เปลี่ยนกรรมวิธีและค่าต่าง ๆ ไปเรื่อย ๆ ผลที่สุดเหลือค่าเดียวคือ ค่าไอดีของปุ่ม Cancel ที่ผมไม่นึกถึง เลยลองเปลี่ยนเป็น 100 ได้ผลครับ

3. จัดฟังก์ชันยุติการพิมพ์ ซึ่งผม

```
(* ------------ Abort Function and Dialog ----------------- *)

VAR   usrstop : Boolean;  hAbortDlg : HWnd;
      abProcPtr : TFarProc;

PROCEDURE  AbortDlgC.SetupWindow;
BEGIN   SetFocus(HWindow)  END;


PROCEDURE  AbortDlgC.wmCommand(var msg : TMessage);
BEGIN
  usrstop := True
END;


FUNCTION  AbortProc(prnDC : HDC; cde : Integer) : Bool; export;
VAR  msg : TMsg;
BEGIN
  while  (not usrstop) and PeekMessage(msg,0,0,0,pm_ReMove)  do
    if  not IsDialogMessage(hAbortDlg, msg)  then
      begin  TranslateMessage(msg);  DispatchMessage(msg)  end;
  AbortProc := not usrstop;
END;


PROCEDURE  DispEscError(escres : Integer);
VAR   errtxt : PChar;
BEGIN
  escres := escres or $FFF8;
  case  escres of
   SP_OUTOFDISK    : errtxt := 'Not enough disk space for spooling.';
   SP_OUTOFMEMORY  : errtxt := 'Not enough memory for spooling.';
   SP_USERABORT    : errtxt := 'User terminated the job.';
  else
    errtxt := 'General error.';
  end;  { case }
  MessageBox(Application^.MainWindow^.HWindow,
             errtxt,'PRINT ERROR', mb_IconExclamation or mb_OK);
END;
```

ลิสติงที่ 1.3




รูปที่ 7
ไดอะล็อกบ็อกซ์
ยุติการพิมพ์

ให้ชื่อว่า AbortProc เพื่อตรวจจับการยุติการพิมพ์ของผู้ใช้โดยตรงที่ตัวแปร usrstop แล้วแจ้งให้ระบบ GDI ทราบ โดยที่ GDI จะเรียกใช้ฟังก์ชันนี้ และฟังก์ชันนี้จะต้องให้ความเป็นจริง เมื่อผู้ใช้ยังให้พิมพ์ตามปกติ และเป็นเท็จเมื่อผู้ใช้ยุติการพิมพ์ (เลือกปุ่ม Cancel ในไดอะล็อกบ็อกซ์) ในโปรแกรมของ AboutProc จึงให้ดำเนินการดังนี้ คือ

- เมื่อเป็นการพิมพ์ตามปกติให้ถอนเมสเสจตามคิว ถ้าหากมี

- ถ้าไม่ใช่เป็นเมสเสจถึงไดอะล็อกบ็อกซ์ ให้ตีความและส่งไปให้สายทางตามปกติ (คือ ทำหน้าที่แทน TheApplC ในช่วงนี้)

- ให้ค่าตนเองเป็นจริงหรือเท็จตรงข้ามกับค่าของ usrstop

แต่ฟังก์ชันนี้ให้ตรวจว่าเป็นเมสเสจถึงไดอะล็อกบ็อกซ์หรือไม่ คือ IsDialog Message จะเป็นผู้แปลงค่าคีย์ เช่นเมื่อผู้ใช้กดคีย์ enter จะแปลงเป็น wm__Command มีค่าไอดีเป็น id__OK ส่งให้แก่ไดอะล็อกบ็อกซ์โดยตรง

4. จัดโปรแกรมเพื่อดำเนินการในเรื่องต่าง ๆ ต่อไปนี้ ซึ่งโปรแกรมในส่วนนี้ผมจะนำไปกล่าวในหัวข้องานพิมพ์ คือ

- ให้กำเนิดไดอะล็อกบ็อกซ์ในแบบโมเดลเลสส์ไดอะล็อกบ็อกซ์

- เชื่อมต่อฟังก์ชันยุติการพิมพ์เข้ากับระบบ GDI

- จัดโปรแกรมสำหรับรับทราบการยุติการพิมพ์ของผู้ใช้คือ ฟังก์ชันยุติการพิมพ์เป็นตัวแจ้งให้ GDI ทราบ และ GDI จะแจ้งให้แอพพลิเคชันทราบอีกทอดหนึ่ง ตามค่าส่งกลับของฟังก์ชัน Escape และถ้ามีข้อขัดข้องก็จะแจ้งมาทางฟังก์ชันนี้เช่นกัน ซึ่งเราจะต้องนำไปดำเนินการ 2 เรื่องคือ ยุติการพิมพ์กับแจ้งให้ผู้ใช้ทราบ ในด้านการแจ้งให้ผู้ใช้ทราบได้ทำโปรแกรมย่อย DispEscError (อยู่ในลิสติง 1.3) ไว้ให้เรียกใช้ ค่าของ escres ดังที่เห็นในโปรแกรม คือค่าที่ส่งกลับมาทาง Escape

## โปรแกรมงานพิมพ์ : สกุล PrinterC

สำหรับโปรแกรมที่เกี่ยวกับงานพิมพ์ที่เหลือได้ทำเป็นสกุล PrinterC ซึ่งดูจะไม่ตรงกับจุดประสงค์ของการสร้างสกุล ในการเขียนโปรแกรมแบบโอโอพีที่ให้สร้างสกุลเพื่อที่จะใช้อินสแตนซ์ของสกุลนั้นหลายตัว

ในกรณีที่จะใช้อินสแตนซ์ของสกุล PrinterC ได้ครั้งละ 1 ตัวเท่านั้น เพราะมีตัวแปรที่เป็นส่วนกลางอยู่ ที่ผมทำเป็นสกุลก็เพื่อจะรวบรวมข้อมูลและโปรแกรมที่เกี่ยวข้องกันไว้ด้วยกัน และต้องการให้มีขั้นตอนของการใช้คือ เริ่มต้นด้วย Init และจบด้วย Done

ที่เพิ่มขึ้นมาคือ ในการพิมพ์งานแต่ละงานให้ขึ้นด้วย StartADoc และจบด้วย EndADoc ในระหว่างที่เขียนโปรแกรมอาจสั่งงานพิมพ์ใดก็ได้ กับได้จัดกิจกรรมของงานพิมพ์ที่มีเป็นประจำ เช่น ให้พิมพ์เท็กซ์ไฟล์ เป็นต้น ดังการกำหนดสกุลในลิสติง 1.2 มีรายละเอียดตามที่จะกล่าวต่อไปนี้ คือ

### ฟิลด์ข้อมูล

ได้กำหนดตัวแปรเป็นฟิลด์ข้อมูลไว้สำหรับเก็บค่าต่าง ๆ โดยค่าที่เป็นระยะเกือบทั้งหมด จะใช้ในการพิมพ์ข้อความและทับเป็นจุดภาพ คือ

- abDlgPtr เก็บค่าพอยน์เตอร์ไปยังไดอะล็อกบ็อกซ์ยุติการพิมพ์

- อะเรย์ margs เก็บค่าขอบบน ขอบซ้าย ขอบล่าง และระยะห่างระหว่างบรรทัด ซึ่งจะให้ค่าเป็นตัวอย่างไว้ และให้ผู้ใช้เปลี่ยนแปลงได้

- pgewid และ pgehi คือความกว้างและความสูงของกระดาษตามที่ผู้ใช้ติดตั้ง เป็นค่าที่จะใช้ในการพิมพ์ภาพบิตแม็พ (และมีอยู่ 2 ค่านี้เท่านั้น) ตามที่จะกล่าวในหัวข้อ สรุปงานพิมพ์

- botlne เก็บค่าบรรทัดสุดท้ายพับจากขอบกระดาษด้านบน (กำหนดเป็นตัวแปรไว้เพื่อไม่ต้องคำนวณบ่อย ๆ)

- lnehi เป็นความสูงของบรรทัดได้จากความสูงของตัวอักษร บวกกับระยะห่างระหว่างบรรทัด

- curlne เก็บค่าบรรทัดปัจจุบันเพื่อการพิมพ์ข้อความเป็นบรรทัดต่อเนื่องกันไป

- prnDC คือ พรินต์ดีซี เทียบได้กับกระดาษพิมพ์ และในบางแห่งผมจะใช้คำนี้

- prnok แสดงสถานะว่าถ้าเป็นจริงให้พิมพ์ได้ตามปกติ

### เตรียมกระดาษพิมพ์

เช่นเดียวกับการพิมพ์ที่ต้องเตรียมกระดาษพิมพ์ ในกรณีนี้คือ เตรียมพรินต์ดีซีตามที่เขียนไว้ในโปรแกรม Init ของสกุลนี้ในลิสติง 1.4 ซึ่งได้จัดให้มีการดำเนินการดังนี้ คือ

- ขอรายละเอียดของเครื่องพิมพ์มาให้กำเนิดพรินต์ดีซีแก่ prnDC

- หากไม่มีข้อขัดข้อง จะให้ค่ากลางของขอบต่าง ๆ ส่งไปให้ DoSelting ดำเนินการในรายละเอียดให้ prnOK เป็นจริงเพื่อแสดงสถานะปกติ

- หากเตรียมพรินต์ดีซีไม่ได้ จะแจ้งให้ผู้ใช้ทราบ และให้ค่าที่ทำให้ดำเนินการพิมพ์ต่อไปไม่ได้

```
CONSTRUCTOR  PrinterC.Init(hWndd : HWnd);
VAR   device,driver,port : PChar;  dc : HDC;
      buf : array[0..80] of Char;
BEGIN
  GetProfileString('windows','device', ',,', buf, SizeOf(buf));
  port := GetToken(@buf, ',');  driver := GetToken(buf, ',');
  device := @buf;

  prnDC := CreateDC(driver, device, port, nil);
  if  prnDC <> 0  then
    begin
    margs[lftt] := 20;  margs[topp] := 100;
    margs[bott] := 100;  margs[lnespcc] := 4;
    prnok := True;  DoSetting;
    end
   else
     begin
     prnok := False;
     MessageBox(hWndd,'Printer initialization failed',
                  'Error',mb_IconExclamation or mb_ok);
     end;
END;



DESTRUCTOR  PrinterC.Done;
BEGIN
  if  prnDC <> 0  then  DeleteDC(prnDC);
END;
```

ลิสติงที่ 1.4

```
FUNCTION   PrinterC.StartADoc(docname : PChar) : Boolean;
VAR   escres : Integer;
BEGIN
  usrstop := False;
  abDlgPtr := AbortDlgCP(Application^.MakeWindow(
     New(AbortDlgCP, Init(Application^.MainWindow, abdlgres))));
  hAbortDlg := abDlgPtr^.HWindow;
  abProcPtr := MakeProcInstance(@AbortProc, HInstance);
  escres := Escape(prnDC, SetAbortProc, 0, abProcPtr, nil);
  if  escres >= 0  then
    escres := Escape(prnDC, StartDoc, StrLen(docname), docname,nil);
  if  escres < 0  then
    begin StartADoc := False;
          DispEscError(escres);  prnok := False  end
  else  StartADoc := True;
END;


FUNCTION   PrinterC.NewPage : Boolean;
VAR  escres : Integer;
BEGIN
  escres := Escape(prnDC, NewFrame, 0, nil, nil);
  if  escres < 0  then
    begin  DispEscError(escres);  prnok := False  end;
  curlne := margs[topp];  NewPage := escres >= 0;
END;


PROCEDURE  PrinterC.EndADoc;
VAR  escres : Integer;
BEGIN
  if  prnok and (curlne > margs[topp])  then  NewPage;
  escres := Escape(prnDC, EndDoc, 0, nil, nil);
  if  abDlgPtr <> nil  then
    begin  abDlgPtr^.CloseWindow;  abDlgPtr := nil  end;
  if  abProcPtr <> nil  then
    begin  FreeProcInstance(@AbortProc);  abProcPtr := nil  end;
END;


FUNCTION  PrinterC.CanPrintBMPs : Boolean;
BEGIN
  CanPrintBMPs :=
    GetDeviceCaps(prnDC, RasterCaps) and rc_BitBlt = rc_BitBlt;
END;
```

ลิสติงที่ 1.5

## เริ่มงานพิมพ์

การที่ผมแยกโปรแกรมตามที่จะกล่าวในหัวข้อนี้ออกมาจาก Init ก็เพื่อเปิดโอกาสให้ผู้ใช้ปรับแต่งค่าขอบต่าง ๆ ได้ เช่น มีรายการ Set Margin ให้ผู้ใช้เลือกต่อเมื่อผู้ใช้ต้องการพิมพ์ เช่น ให้เลือกรายการ Print จึงให้เริ่มงานพิมพ์ตามโปรแกรมของกิจกรรม StartADoc ในลิสติง 1.5

โดยให้ผ่านค่า docname เป็นชื่อเอกสารที่จะพิมพ์ (ชื่อนี้จะไปปรากฏใน Print Manager) เพื่อดำเนินการต่อไป คือ

- แสดงไดอะล็อกบ็อกซ์ยุติการพิมพ์ เป็นโมเดลเลสส์ไดอะล็อกบ็อกซ์ คือ ใช้คำสั่ง MakeWindow หากให้เป็นโมเดลไดอะล็อกบ็อกซ์ คือ ใช้คำสั่ง ExecDialog แทน โปรแกรมจะติดอยู่ตรงนี้จนกว่าผู้ใช้จะยุติการทำงานในไดอะล็อกบ็อกซ์ และนี่เป็นตัวอย่างที่แสดงข้อแตกต่างของไดอะล็อกบ็อกซ์ทั้ง 2 ประเภทนี้อย่างเห็นเด่นชัด

- ต่อฟังก์ชันยุติการพิมพ์เข้ากับระบบ GDI ของวินโดวส์ โดยทำให้เป็นอินสแตนซ์แล้วต่อเชื่อมด้วยฟังก์ชัน Escape

- ถ้าไม่มีปัญหาจะเริ่มงานพิมพ์ด้วยเอสเคป StartDoc

## ขึ้นหน้าใหม่

หลังจากที่ให้เริ่มงานพิมพ์แล้ว เป็นหน้าที่ของโปรแกรมที่เรียกใช้สกุลนี้จะส่งเนื้อหาที่จะให้พิมพ์แก่พรินต์ดีซี 1 หน้า แล้วให้ขึ้นหน้าใหม่ด้วยกิจกรรม NewPage ซึ่งจะเป็นผู้ส่งเอสเคป NewFrame ไปให้แก่ GDI เป็นการส่งงานพิมพ์ 1 หน้าไปให้แก่พรินต์แมนเนเจอร์

และจุดนี้เป็นจุดที่เหมาะสมที่สุดที่จะตรวจว่าผู้ใช้ยุติการพิมพ์หรือไม่

มีข้อขัดข้องหรือไม่ โดยตรวจดูค่าที่ฟังก์ชัน Escape ส่งมาใน escres หากมีเหตุจะให้ prnok เป็นเท็จ เพื่อใช้ในการยุติการพิมพ์ ดังจะเห็นในโปรแกรมต่อไป

## จบงานพิมพ์

หมายถึงการจบงานพิมพ์ของเอกสารแต่ละชิ้น ซึ่งอาจให้พิมพ์หน้าเดียวหรือหลายหน้าก็ได้ เสร็จแล้วต้องจบงานพิมพ์ด้วย EndADoc ซึ่งผมได้ออกแบบไว้ว่า เมื่อใช้ StartADoc แล้วต้องใช้ EndADoc คู่กันเสมอ เพื่อ

- ให้พิมพ์เอกสารอันอาจตกค้าง
- ส่งเอสเคป EndDoc แจ้งให้ Print Manager ทราบ
- ปิดไดอะล็อกบ็อกซ์ยุติการพิมพ์
- ตัดทางติดต่อของฟังก์ชันยุติการพิมพ์กับระบบ GDI

เป็นการจบงานพิมพ์ของงานหนึ่ง ๆ อย่างแท้จริง หากจะพิมพ์เอกสารอื่นต่อไปจะต้องเริ่มตั้งแต่ StartADoc ใหม่

## เลิกพิมพ์

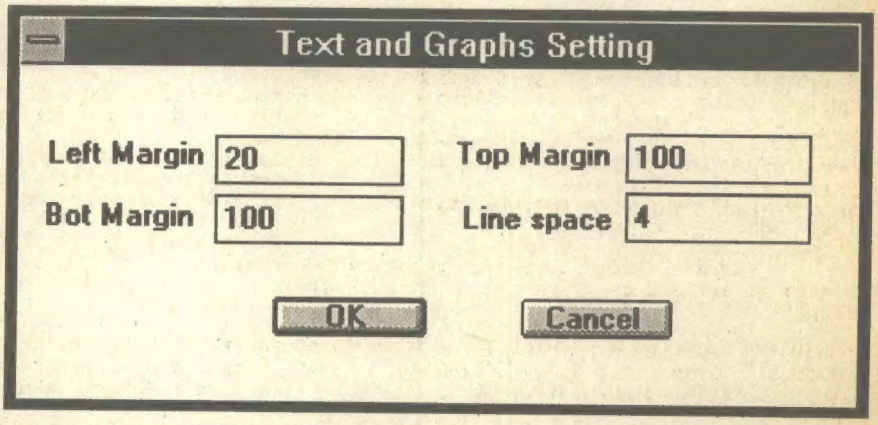
เทียบเท่ากับปิดเครื่องพิมพ์ แต่เป็นเรื่องของสกุลครับ คือเมื่อ Init แล้วต้อง Done สำหรับสกุลนี้เนื่องจากได้ให้กำเนิดพรินต์ดีซีไว้ใน Init จึงต้องยกเลิกใน Done (อยู่ในลิสติง 1.4)

## ตั้งขอบในการพิมพ์

เพื่อให้สามารถตั้งขอบในการพิมพ์ได้ด้วย จึงได้เตรียมกิจกรรมไว้ 2 กิจกรรม คือ SetMargins ในลิสติง 1.6 เพื่อให้โปรแกรมที่ใช้สกุลนี้ส่งค่าที่ต้องการมาให้ กับกิจกรรม PaperSetting เพื่อให้ผู้ใช้ตั้งค่า โดยแสดงไดอะล็อกบ็อกซ์ของสกุล PrnSetDlgC ดังรูปที่ 8 (โปรแกรมของสกุลนี้ผมขอนำไปกล่าวในตอนท้าย)

```
PROCEDURE  PrinterC.SetMargins(lft,top,bot,dif : Integer);
BEGIN
  margs[lftt] := lft;  margs[topp] := top;
  margs[bott] := bot;  margs[lnespcc] := dif;
  DoSetting;
END;


PROCEDURE  PrinterC.PaperSetting;
VAR   dlgp : PDialog;
BEGIN
  dlgp := New(PrnSetDlgCP, Init(nil, txtdlgres,
                @margs,id_LeftMar,id_LineSpace));
  if  Application^.ExecDialog(dlgp) = id_OK  then
    DoSetting;
END;



PROCEDURE  PrinterC.DoSetting;
VAR    TM : TTextMetric;
BEGIN
  GetTextMetrics(prnDC, TM);
  if  margs[lnespcc] = 0  then
    begin  margs[lnespcc] := TM.tmHeight div 4;
           if  margs[lnespcc] = 0  then  margs[lnespcc] := 1  end;
  lnehi := TM.tmHeight + margs[lnespcc];
  pgewid := GetDeviceCaps(prnDC, HorzRes);
  pgehi  := GetDeviceCaps(prnDC, VertRes);
  botlne := pgehi - margs[bott] - lnehi;
  curlne := margs[topp];
END;
```

ลิสติงที่ 1.6




รูปที่ 8 ไดอะล็อกบ็อกซ์ของสกุล PrnSetDlgC มีค่าไอดีของกรอบอีดิตตามชื่อที่ตั้งกำกับไว้ ดังนี้คือ

Left Margin = 101 Top Margin = 102

Bot Margin = 103 Line space = 104

เมื่อได้ค่ามาแล้วทั้ง SetMargins และ PaperSetting (รวมทั้ง Init ด้วย) จะเก็บลงอะเรย์ margs แล้วให้ DoSetting ดำเนินการในรายละเอียด คือ

- ถ้าให้ค่าช่องว่างระหว่างบรรทัดเป็น 0 จะหมายถึงต้องการให้มีช่องว่างระหว่างบรรทัดเป็น 1/4 ของความสูงของตัวอักษร
- ให้ความสูงของบรรทัดเท่ากับความสูงของตัวอักษร บวกด้วยช่องว่างระหว่างบรรทัดแก่ lnehi เพื่อใช้ในการขึ้นบรรทัดใหม่
- เก็บค่าความกว้างและความสูงของกระดาษไว้ใน pgewid กับ pgehi

หากต้องการใช้ในการพิมพ์ภาพกราฟิก รวมทั้งบิตแม็พ

- หาตำแหน่งของบรรทัดล่างให้แก่ Botlne และ curlne ชี้ที่บรรทัดบน

## กิจกรรมสนับสนุนงานพิมพ์

ได้เขียนโปรแกรมสำหรับสนับสนุนงานพิมพ์ข้อความ (text) ไว้จำนวนหนึ่งตามลิสติง 1.7 คือ

### PrintLine

เป็นกิจกรรมที่ต้องการให้เรียกใช้ในวงรอบ เพื่อการพิมพ์ข้อความครั้งละ 1 บรรทัด แล้วขึ้นบรรทัดใหม่ด้วย NewLine ซึ่ง NewLine จะตรวจดูว่าสุดหน้าหรือยัง ถ้าสุดหน้าจะให้ขึ้นหน้าใหม่ด้วย NewPage เป็นการส่งข้อความออกพิมพ์ด้วยตามที่กล่าวมา

### PrintFile

เพื่อการพิมพ์ข้อความ 1 ไฟล์ ด้วยการอ่านข้อความมาครั้งละ 1 บรรทัด ให้ PrintLine ดำเนินการพิมพ์ในวงรอบ จะเห็นว่าให้ตรวจ prnok ด้วย ถ้าผิดปกติให้ยุติการพิมพ์ (ก็เพิ่งครบถ้วนขบวนความของการยุติการพิมพ์ตรงนี้ครับ ว่าจะยุติงานพิมพ์ได้อย่างไร)

### PrintEditor และ PrintEdBlock

เพื่อพิมพ์ข้อความในเอดิเตอร์ (อินแสตนซ์ของสกุล TEdit) หากต้องการพิมพ์ทั้งหมดให้ใช้ PrintEditor หากต้องการพิมพ์บางส่วนที่ผู้ใช้หมายกลุ่มไว้ให้ใช้ PrintEdBlock อยู่ในลิสติงที่ 1.8

## ไดอะล็อกบ็อกซ์ตั้งขอบ

โปรแกรมของไดอะล็อกบ็อกซ์ที่ให้ผู้ใช้ตั้งขอบในการพิมพ์ตามรูปที่ 7 คือสกุล PrnSetDlgC อยู่ในลิสติง 1.8 ขอให้สังเกตด้วยว่ามี END. ของยูนิต เพราะผมป้อนไว้ท้ายสุด เป็นการใช้สกุล EditNumC ที่ผมสร้างไว้ในยูนิต UFiDlg เพื่อการป้อนเลขจำนวน โดยที่

1. ต้องสร้างรีซอร์สของไดอะล็อกบ็อกซ์ตามรูปที่ 7 ให้ชื่อว่า SETTEXT_DLG เก็บลงไฟล์ UPRINT.

```
PROCEDURE  PrinterC.PrintLine(txt : PChar);
BEGIN
  TextOut(prnDC, margs[lftt], curlne, txt, StrLen(txt));
  NewLine;
END;

PROCEDURE  PrinterC.NewLine;
BEGIN
  curlne := curlne + lnehi;
  if  curlne > botlne then  NewPage;
END;


PROCEDURE  PrinterC.PrintLineAt(xx,yy : Integer; txt : PChar);
BEGIN
  TextOut(prnDC, xx, yy, txt, StrLen(txt));
END;


PROCEDURE  PrinterC.PrintFile(finme : PChar);
VAR   datfi : Text;  astr : String;
      txt : array[0..255] of Char;
BEGIN
  astr := StrPas(finme);  Assign(datfi,astr);
  {$I-} Reset(datfi);  {$I+}
  if  IOresult <> 0  then
    begin  DispErr(2);  Exit  end;
  while (not EOF(datfi)) and prnok  do
    begin  ReadLn(datfi, astr);  StrPCopy(txt, astr);
           PrintLine(txt)  end;
  Close(datfi);
END;
```

ลิสติงที่ 1.7

```
PROCEDURE  PrinterC.PrintEditor(edtr : PEdit);
VAR   numlne,nlne,len : Integer;  lnebuf : PChar;
BEGIN
  numlne := edtr^.GetNumLines;  nlne := 0;
  while  (nlne < numlne) and prnok  do
    begin
    len := edtr^.GetLineLength(nlne) + 1;
    GetMem(lnebuf, len);
    if  len > 1  then  edtr^.GetLine(lnebuf, len, nlne)
    else  lnebuf^ := #0;
    PrintLine(lnebuf);  FreeMem(lnebuf, len);  inc(nlne);
    end;
END;


PROCEDURE  PrinterC.PrintEdBlck(edtr : PEdit);
VAR  sttpos,endpos,sttlne,endlne,lne,si : Integer;
     lnebuf : Pchar;  membuf : array[0..255] of Char;
BEGIN
  edtr^.GetSelection(sttpos,endpos);
  if  sttpos = endpos  then  Exit;
  sttlne := edtr^.GetLineFromPos(sttpos);
  endlne := edtr^.GetLineFromPos(endpos);
  lnebuf := @membuf;
  for  lne := sttlne to endlne  do
    begin
    if  not prnok  then  Exit;
    si := edtr^.GetLineLength(lne) + 1;
    if  si > 1  then  edtr^.GetLine(lnebuf, si, lne)
    else  lnebuf^ := #0;
    PrintLine(lnebuf);
    end;
END;
```

ลิสติงที่ 1.8

```
(* ------------ PrnSetDlgC Method  ------------ *)

CONSTRUCTOR  PrnSetDlgC.Init(apar : PWindowsObject; atitle : PChar;
                             dataaray : MarginP; fstid,lstid : Integer);
VAR   id : MargIndT;
BEGIN
  TDialog.Init(apar,atitle);
  for  id := lftt to lnespcc  do
    ednums[id] := New(EditNumCP, InitResource(@self,Ord(id)+fstid,'i'));
  datary := dataaray;  idfst := fstid;  idlst := lstid;
END;


PROCEDURE  PrnSetDlgC.SetupWindow;
VAR  id : MargIndT;
BEGIN
  TDialog.SetupWindow;
  for  id := lftt to lnespcc  do
    ednums[id]^.SetNum(datary^[id], 0);
  EnableKBHandler;   SetFocus(ednums[lftt]^.HWindow);
END;


PROCEDURE  PrnSetDlgC.Ok(var msg : TMessage);
VAR   id : MargIndT;
BEGIN
  for  id := lftt to lnespcc  do
    ednums[id]^.GetNum(datary^[id]);
  TDialog.OK(msg);
END;


END.
```

ลิสติงที่ 1.9

RES

2. การผ่านค่า ขอให้ดูในโปรแกรมของกิจกรรม PaperSetting ในลิสติง 1.6

3. ในสกุลนี้ได้ดำเนินการดังนี้ คือ

- ทำให้กรอบอีดิตทุกกรอบเป็นออปเจ็กต์สกุล EditNumC ด้วย Init-Resource

- ใน SetupWindow ได้ให้ค่าที่ผ่านมาทางอะเรย์ dataarray แก่กรอบอีดิตทุกกรอบ เพื่อนำออกแสดง

- เมื่อผู้ใช้เลือกปุ่ม OK จะอ่านค่าจากกรอบอีดิตทุกกรอบให้แก่อะเรย์ dataarray ส่งกลับไป

## สรุปงานพิมพ์

ตามที่ผมกล่าวมานี้เป็นเรื่องของการพิมพ์ข้อความเท่านั้น เพราะการพิมพ์ภาพกราฟิกรวมทั้งบิตแม็พง่ายกว่ากันมากครับ จึงขอสรุปการใช้สกุล PrinterC เพื่อการพิมพ์ดังนี้

1. ใช้ Init เพื่อเตรียมพรินต์ดีซี

2. หากต้องการเซตขอบกระดาษให้ใช้ SetMargin หรือ PaperSetting แล้วแต่กรณี

3. ใช้ StartADoc ขึ้นต้นงานพิมพ์

4. ส่งเอกสารไปให้พรินต์ดีซี ตรงนี้ครับที่แตกต่างกัน คือ

- ถ้าเป็นข้อความให้ส่งครั้งละ 1 บรรทัดด้วย TextOut จนหมดหรือเต็มหน้า

- ถ้าเป็นกราฟิก ให้ใช้ฟังก์ชันทางกราฟิกดังตัวอย่างในรูปที่ 2

- ถ้าเป็นบิตแม็พ ให้ใช้ PrnDsp-BMPs ตามที่ได้เขียนโปรแกรมไว้แล้ว

5. ใช้ NewPage เพื่อส่งเอกสารออกพิมพ์ และเป็นการขึ้นหน้าใหม่ ซึ่งถ้ามีเอกสารหลายหน้าจะต้องทำข้อ 4 และ 5 ซ้ำใหม่ และถ้าเป็นข้อความได้เตรียมกิจกรรมสนับสนุนงานพิมพ์ไว้แล้ว ซึ่งจะเป็นการรวมข้อ 4 และ 5 เข้าด้วยกัน

6. ใช้ EndADoc เพื่อแจ้งให้พรินต์แมนเนเจอร์ทราบว่าจบเอกสาร 1 แฟ้ม หากมีเอกสารอื่นอีกให้ดำเนินการตามข้อ 3 ถึง 6 ซ้ำ

7. ใช้ Done เพื่อจบการทำงานกับพรินเตอร์

## โปรแกรมทั้งหมด

ขอให้ป้อนโปรแกรมตามลิสติง 1.1 ถึง 1.8 โดยขึ้นต้นด้วย 1.1 ตามด้วย 1.2 นอกนั้นจะป้อนสลับกันอย่างไรก็ได้ โดยให้ลิสติง 1.8 อยู่ด้วย หรือไม่ก็นำ END. ไปไว้ท้าย ไว้ในไฟล์ UPRINT1.PAS

## โปรแกรมทดสอบ

โปรแกรมทดสอบสกุล PrintC ในด้านการพิมพ์ข้อความจากเท็กซ์ไฟล์อยู่ในลิสติง 2 ซึ่งจะแสดงไดอะล็อก

บ็อกซ์ให้ผู้ใช้เลือกไฟล์นำมาพิมพ์

หากต้องการพิมพ์ภาพบิตแม็พให้ใช้โปรแกรมในลิสติง 3 ซึ่งผมแสดงแต่โปรแกรมของกิจกรรม cmPrint ขอให้ใช้โครงโปรแกรมของลิสติง 2 ให้ผู้ใช้เลือกไฟล์มาพิมพ์เช่นกัน แต่ต้องเป็นไฟล์ .BMP

## การพิมพ์ผ่านดอส

บนวินโดวส์และสำหรับเทอร์โบปาสคาล นอกจากจะให้พิมพ์ผ่านระบบ GDI ของวินโดวส์แล้ว ยังสามารถให้พิมพ์ผ่านดอสคือ ไม่ผ่านระบบ GDI ไม่อยู่ในความควบคุมของพรินต์แมนเนเจอร์ โดยใช้คำสั่งของเทอร์โบปาสคาลดังเช่นเมื่ออยู่บนดอสตามโปรแกรมในลิสติง 4

การพิมพ์ในแบบนี้จึงขึ้นอยู่กับเครื่องพิมพ์ คือต้องตั้งทุกอย่างที่เครื่องพิมพ์ และเมื่อพิมพ์จะต้องไม่มีงานพิมพ์อื่นที่ผ่านระบบ GDI คือ ที่พรินต์แมนเนเจอร์กำลังดำเนินการอยู่ เพราะจะทำให้สับสน แต่ผมก็ชอบพิมพ์ด้วยวิธีนี้มากกว่าเพราะพิมพ์ได้เร็วกว่า

ขอให้นำโปรแกรมในลิสติง 4 ไปแทรกไว้ในยูนิต UPrint1 ตามที่ผมได้แสดงส่วนหัวไว้แล้ว ในส่วนอินเตอร์เฟสและยังมีอีก 3 โพรซีเดอร์ครับ

- DosPrnBlck พิมพ์กลุ่มข้อความ
- WrtBlck เก็บกลุ่มข้อความลงไฟล์
- ReadBlck อ่านไฟล์มาแทรกตรงเคอร์เซอร์

ขอให้ทดลองเขียนดูครับ โดยเปรียบเทียบจากโปรแกรมลิสติง 4 กับในลิสติง 1.7

```
PROGRAM   PrintTextFile;

USES   WinTypes,WinProcs,WObjects,Strings,ufidlg,uprint1;
CONST  appname  = 'dlgwnd';
       thetitle = 'TEST TDlgWindow';

TYPE
  WndProCP = ^WndProC;
  WndProC = OBJECT(TWindow)
 constructor  Init(apar : PWindowsObject; atitle : PChar);
 procedure  wmRButtonDown(var msg : TMessage);
              virtual wm_First + wm_RButtonDown;
            END;


(* --------------- WndProC Routine --------------- *)

CONSTRUCTOR  WndProC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN
  TWindow.Init(apar,atitle);
END;


PROCEDURE  WndProC.wmRButtonDown(var msg : TMessage);
VAR    finme : array[0..pathlen] of Char;
       exts  : array[0..29] of Char;  aprn : PrinterCP;
BEGIN
  StrCopy(exts,'*.*,*.me,*.txt,*.pas');
  if Application^.ExecDialog(New(SingleFiDlgCP,
       Init(nil,
         StrCopy(finme, '*.pas'), exts)))
             = id_Cancel  then  Exit;

  New(aprn,Init(HWindow));
  if  aprn^.prnok  then
    begin
    if  aprn^.StartADoc(finme)  then
      aprn^.PrintFile(finme);
    aprn^.EndADoc;
    end;
  Dispose(aprn,Done);
END;

(* ------------------  TheAppC Class  ---------------- *)

TYPE
  TheAppC = OBJECT(TApplication)
    procedure  InitMainWindow; virtual;
               END;


PROCEDURE  TheAppC.InitMainWindow;
BEGIN
  MainWindow := New(WndProCP, Init(nil,thetitle));
END;

(* --------------------  MAIN PROGRAM  ---------------- *)

VAR   app :  TheAppC;
  app.Init(appname);
  app.Run;
  app.Done;
END.
```

ลิสติงที่ 2

## ฝากทดลอง

สำหรับท่านที่สนใจในบริการซอฟต์เมล์อยู่แล้ว จะมีโปรแกรม Print Job1 ในไฟล์ PRNJOB1.EXE ติดมาด้วย เป็นโปรแกรมที่ผมเขียนขึ้นเพื่อใช้ส่วนตัว ยังไม่พร้อมที่จะนำมาเสนอ แต่สำหรับท่านที่มีเวลาพอขอความกรุณาช่วยทดลองรันหาข้อขัดข้องและถ้าต้องการเปลี่ยนแปลงเพิ่มเติมสิ่งใด กรุณาแจ้งไปด้วยจะขอบคุณมากครับ

```
PROCEDURE  WndProC.wmRButtonDown(var msg : TMessage);
VAR    finme : array[0..pathlen] of Char;
       exts  : array[0..29] of Char;  aprn : PrinterCP;
       hBMP : HBitmap;  ww,hh : LongInt;
BEGIN
  StrCopy(exts,'*.*,*.bmp');
  if Application^.ExecDialog(New(SingleFiDlgCP,
       Init(nil,
         StrCopy(finme, '*.pas'), exts)))
            = id_Cancel  then  Exit;

  hBMP := LoadBitMapFile(finme, ww, hh);

  New(aprn,Init(HWindow));
  if  aprn^.prnok  then
    begin
    if  aprn^.StartADoc(finme)  then
      PrnDspBmps(HWindow, hBMP, aprn^.prnDC,
                 100,100,ww,hh, false,
                 0,0,ww,hh);
    aprn^.NewPage;  aprn^.EndADoc;
    end;
  Dispose(aprn,Done);  DeleteObject(hBMP);
END;
```

ลิสติงที่ 3

```
PROCEDURE  DosPrnTxtFi(finme : PChar);
VAR   lne : String;  datfi,prnt : Text;  res : Integer;
BEGIN
  Assign(datfi,finme);
  {$I-} Reset(datfi);  {$I+}   res := IOresult;
  if  res <> 0  then  begin  DispErr(res);  Exit  end;
  Assign(prnt,'PRN');  ReWrite(prnt);
  while  not EOF(datfi)  do
    begin  ReadLn(datfi, lne);  WriteLn(prnt, lne)  end;
  WriteLn(prnt, #12);  { form feed }
  Close(datfi);  Close(prnt);
END;
```

ลิสติงที่ 4

เมื่อรันโปรแกรมนี้จะแสดงเมนูมีรายการที่คุ้นอยู่แล้ว ที่เพิ่มมาคือ

1. ในรายการ File มีรายการ

- Write Block เพื่อเก็บกลุ่มข้อความลงไฟล์ เช่นเดียวกับที่ใช้ Ctrl + K, W

- Read Block อ่านไฟล์มาแทรก เช่นเดียวกับ Ctrl + K, R

2. ในรายการ Print มีรายการ

- Print พิมพ์ข้อความตามที่โหลดด้วย File/Open

- Print Block พิมพ์กลุ่มข้อความ

- Print Spool ให้เลือกไฟล์มาพิมพ์ได้หลายไฟล์

- Set Paper เพื่อตั้งขอบต่าง ๆ ดังกล่าวมา

- รายการ Dos Print, Dos Print Block และ Dos Print Spool เป็นการพิมพ์ผ่านดอสครับ

## ซอฟต์เมล์

ในซอฟต์เมล์ของฉบับนี้จะมีไฟล์อยู่หลายไฟล์ เพื่อไม่ให้สับสนจะขอจำแนกดังนี้ ครับ

- ไฟล์ pict1.pas และ pict2.pas เป็นไฟล์ที่รันได้ตามโปรแกรมในรูปที่ 1 และที่ 2

- ไฟล์ pict4.pas และ pict5.pas เป็นไฟล์ที่รันได้ของรูปที่ 4 และที่ 5 สำหรับผู้ที่ใช้เทอร์โบปาสคาล 1.5

- ไฟล์ ufidlg.pas กับ ufidlg.res เป็นไฟล์ของยูนิต UFiDlg (เป็นไฟล์เดียวกับที่กล่าวในไมโครฯ 91) ซึ่งต้องคอมพายล์เป็นไฟล์ .TPU

- ไฟล์ uprint1.pas กับ uprint.res เป็นไฟล์ตามที่กล่าวในฉบับนี้ ซึ่งต้องคอมพายล์เป็นไฟล์ .TPU เช่นกัน

- ไฟล์ prntxtfi และ prmbmpfi คือไฟล์ตามลิสติง 2 และ 3 (ที่รันได้) เพื่อทดสอบสกุล PrinterC ในยูนิต UPrint1

- ไฟล์ prinjob1.exe คือไฟล์ที่ฝากมาทดลองใช้ครับ

## ท้ายตอน

เรื่องของพรินเตอร์ยังมีอีกครับ เช่น ฟอนต์และการเซตเครื่องพิมพ์ ซึ่งผมต้องขอพักไว้ก่อน ฉบับหน้าจะเป็นการนำยูนิตที่ทำไว้มาใช้ พบกันในเรื่องจับภาพใส่หน้าต่างครับ